Temporal Precedence Checking for Switched Models and its Application to a Parallel Landing Protocol

Parasara Sridhar Duggirala[1], Le Wang[1], Sayan Mitra[1], Mahesh Viswanathan[1], and César Muñoz[2]

[1] University of Illinois at Urbana Champaign,
{duggira3,lewang2,mitras,vmahesh}@illinois.edu
[2] NASA,
cesar.a.munoz@nasa.gov

Abstract. This paper presents an algorithm for checking temporal precedence properties of nonlinear switched systems. This class of properties subsume bounded safety and capture requirements about visiting a sequence of predicates within given time intervals. The algorithm handles nonlinear predicates that arise from dynamics-based predictions used in alerting protocols for state-of-the-art transportation systems. It is sound and complete for nonlinear switch systems that robustly satisfy the given property. The algorithm is implemented in the Compare Execute Check Engine (C2E2) using validated simulations. As a case study, a simplified model of an alerting system for closely spaced parallel runways is considered. The proposed approach is applied to this model to check safety properties of the alerting logic for different operating conditions such as initial velocities, bank angles, aircraft longitudinal separation, and runway separation.

1 Introduction

Dynamic analysis presents a scalable alternative to static analysis for models with nonlinear dynamics. The basic procedure for dynamic safety verification has three building blocks: (a) a simulation engine, (b) a generalization or bloating procedure, and (c) a satisfiability checker. The simulation engine generates validated simulations of the model with some rigorous error bounds. The generalization procedure uses additional model information to over-approximate bounded-time reach sets from the simulations. This additional model information could be, for example, statically computed Lipschitz constants [11], contraction metrics [7] or more general designer-provided annotations [4]. Finally, the approximation is checked by a satisfiability procedure for inferring safety or for iteratively refining its precision. With these three pieces it is possible to design sound and relatively complete algorithms for bounded time safety verification that also scale to moderately high-dimensional models [4].

This paper proposes a new algorithm that extends the reach of the above procedure in two significant ways. First, the new algorithm verifies *temporal* *precedence properties* which generalize bounded safety. A model $\mathcal{A}$ satisfies temporal precedence $P_1 \prec_b P_2$ if along every trajectory of $\mathcal{A}$, for any time at which the predicate P_2 holds, there exists an instant of time, at least b time units sooner, where the predicate P_1 must hold. The key subroutine in the new verification algorithm uses a simulation-based reach set approximation procedure for estimating the time intervals over which the predicates P_1 and P_2 may or must hold. These estimates are constructed so that the algorithm is sound. The algorithm is guaranteed to terminate whenever $\mathcal{A}$ satisfies the given property robustly (relatively complete). That is, not only does every trajectory ξ satisfy $P_1 \prec_b P_2$, but any small time-shifts and value perturbations of ξ also satisfy $P_1 \prec_b P_2$. Such relative completeness guarantees usually have the most precision that one can hope for in any formal analysis of models involving physical quantities.

Secondly, a new approach to checking satisfiability of nonlinear guarantee predicates [8] is proposed. If P_1 and P_2 in the above type of temporal precedence property are in propositional logic or uses linear arithmetic, then existing solvers can efficiently check whether a set of states satisfy them. On the other hand, if they are written as $\exists t > 0, f_p(x,t) > 0$, where f_p is a nonlinear real-valued function, then the options are limited. Quantifier elimination is an expensive option, but even that is feasible only if f_p has a closed form definition of a special form (such as polynomial functions). If f_p is implicitly defined as the solution of a set of ODEs with no analytical solution then quantifier elimination is impossible. This paper provides a sound and relatively complete procedure for checking bounded time guarantee predicates using simulation-based over-approximations of $f_p(x,t)$.

The new algorithm is used in the analysis of an interesting and difficult verification problem arising from a parallel landing protocol. The Simplified Aircraft-based Paired Approach (SAPA) [5] is an advanced operational concept that enables dependent approaches in closely spaced parallel runways. In the presence of blundering aircraft, the SAPA procedure relies on an alerting algorithm called Adjacent Landing Alerting System (ALAS) [10]. ALAS uses linear and nonlinear projections of the current aircraft positions, velocity vectors, and bank angles to detect possible conflicts between the landing aircraft. Given the nonlinear characteristics of the ALAS logic, finding operating conditions under which the SAPA/ALAS protocol satisfy some safety properties is a challenging problem.

This paper presents a simplified model, written as a switched system, of the SAPA/ALAS protocol. The safety properties that are considered on this model state that an alert is issued at least b seconds before an unsafe scenario is encountered. These properties are specified as temporal precedence properties of the form $Alert \prec_b Unsafe$. The proposed verification algorithm is applied to this model to formally check these kinds of safety properties for various aircraft and runway configurations.

2 System Models and Properties

For a vector v in $\mathbb{R}^n$, $|v|$ stands for ℓ^2-norm. Given intervals I,I', the relation $I < I'$ holds iff $\forall v \in I, \forall v' \in I', v < v'$. For a real number b, $I - b = \{v - b | v \in I\}$. Subtraction operation over intervals is defined as, $I - I' = \{v - v' | v \in I, v' \in I'\}$. $I \times I' = \{v \times v' | v \in I, v' \in I'\}$. For $\delta \in \mathbb{R}_{\geq 0}$ and $x \in \mathbb{R}^n$, $B_\delta(x) \subseteq \mathbb{R}^n$ is the closed ball with radius δ centered at x. For a set $S \subseteq \mathbb{R}^n$, $B_\delta(S) = \cup_{x \in S} B_\delta(v)$. For any function $V : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$, given a $\delta > 0$, $B^V_\delta(x) = \{y \mid V(x, y) \leq \delta\}$. For a set $S \subseteq \mathbb{R}^n$, $B^V_\delta(S) = \cup_{x \in S} B^V_\delta(x)$. For a bounded set A, $dia(A) = \sup_{x,y \in A} |x - y|$ denotes the diameter of A.

A real-valued function $\alpha : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is called a *class $\mathcal{K}$ function* if $\alpha(0) = 0$ and α is strictly increasing. It is a *class $\mathcal{K}_\infty$ function* if additionally $\alpha(x) \to \infty$ as $x \to \infty$. For a function $h : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ and a positive real $\delta > 0$, the δ-left shift of h is the function $h_\delta : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ defined as $h_\delta(t) = h(t + \delta)$ for any $t \in \mathbb{R}_{\geq 0}$. A *$\delta$-perturbation* of h is any function $g : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ such that for all t, $|g(t) - h(t)| < \delta$. A *càdlàg* function is a function which is *continuous from the right* and *has a limit from the left* for every element in its domain.

2.1 The Switched System Model

This paper uses the *switch system* formalism [6] for modeling continuous systems. The evolution of an n dimensional switched system is specified by a collection of ordinary differential equations (ODEs) also called as *modes* or *locations* indexed by a set $\mathcal{I}$ and a *switching signal* that specifies which ODE is active at a given point in time. Fixing a switching signal and an initial state, the system is deterministic. Its behavior is the continuous, piece-wise differentiable function of time obtained by pasting together the solutions of the relevant ODEs. The symbol $\mathcal{I}$ represents the set of modes and n represents the dimension of the system with $\mathbb{R}^n$ as state space.

Definition 1. *Given the set of modes $\mathcal{I}$ and the dimension n, a* switched system $\mathcal{A}$ *is specified by the tuple $\langle \Theta, \mathcal{F}, \Sigma \rangle$, with*

(i) $\Theta \subseteq \mathbb{R}^n$*, a compact set of* initial states*,*
(ii) $\mathcal{F} = \{f_i : \mathbb{R}^n \to \mathbb{R}^n\}_{i \in \mathcal{I}}$*, an indexed collection of continuous, locally Lipschitz functions, and*
(iii) Σ*, a set of switching signals, where each $\sigma \in \Sigma$ is a càdlàg function $\sigma : \mathbb{R}_{\geq 0} \to \mathcal{I}$.*

The semantics of $\mathcal{A}$ is defined in terms of its solutions or *trajectories*. For a given initial state $x_0 \in \Theta$ and a switching signal $\sigma \in \Sigma$, the solution or the *trajectory* of the switched system is a function $\xi_{x_0,\sigma} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, such that: $\xi_{x_0,\sigma}(0) = x_0$, and for any $t > 0$ it satisfies the differential equation:

$$\dot{\xi}_{x_0,\sigma}(t) = f_{\sigma(t)}(\xi_{x_0,\sigma}(t)). \tag{1}$$

When clear from context, the subscripts x_0 and σ are dropped from ξ. Under the stated locally Lipschitz assumption of the f_i's and the càdlàg assumption on σ, it is well-known that Equation (1) has a unique solution and that indeed the trajectory ξ is a well-defined function.

A bounded time switching signal can be represented as a sequence $\sigma = m_0, m_1, \ldots, m_k$ where each m_i is a pair in $\mathcal{I} \times \mathbb{R}_+$, with the two components denoted by $m_i.mode$ and $m_i.time$. The sequence define $\sigma(t) = m_i.mode$ for all $t \in [\sum_{j=0}^{i-1} m_j.time, \sum_{j=0}^{i} m_j.time)$. A set of switching signals Σ is represented as a switching interval sequence $S = q_0, q_1, \ldots q_k$, where each q_j is a pair with $q_j.mode \in \mathcal{I}$ and $q_j.range$ is an open interval in $\mathbb{R}_{\geq 0}$. Given a switching interval sequence S, the set $sig(S)$ denotes the set of switching signals $\sigma = m_0, m_1, \ldots, m_k$, such that $m_j.mode = q_j.mode$ and $m_j.time \in q_j.range$. By abuse of notation, a set of switching signals Σ and its finite representation S with $sig(S) = \Sigma$ are used intechangeably. The expression $width(S)$ denotes the size of the largest interval $q_i.range$. The refinement operation of Σ, denoted as $refine(S)$, gives a finite set of switching interval sequences $\mathcal{S}$ such that $\bigcup_{S' \in \mathcal{S}} sig(S') = sig(S)$ and for each $S' \in \mathcal{S}$, $width(S') \leq width(S)/2$.

2.2 Temporal Precedence with Guarantee Predicates

A *predicate* for the switched system $\mathcal{A}$ is a computable function $P : \mathbb{R}^n \to \{\top, \bot\}$ that maps each state in $\mathbb{R}^n$ to either $\top$ (true) or $\bot$ (false). The predicate is said to be satisfied by a state $x \in \mathbb{R}^n$ if $P(x) = \top$. A *guarantee predicate* [8] $P(x)$ is a predicate of the form $\exists t > 0, f_p(x,t) > 0$, where $f_p : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ is called a *lookahead function*. A guarantee predicate holds at a state x if there exists some future time t at which $f_p(x,t) > 0$ holds. Using a quantifier elimination procedure, a guarantee predicate can be reduced to an ordinary predicate without the existential quantifier. However, this is an expensive operation, and more importantly, it is only feasible for restricted classes of real-valued lookahead functions with explicit closed form definitions. Section 3.1 presents a technique to handle guarantee predicates with lookahead functions as solutions to nonlinear ODE. As seen in Section 4, such lookahead functions are particularly useful in designing alerting logics such as ALAS.

Temporal precedence properties are a class of properties specified by a pair of predicates that must hold for any behavior of the system with some minimum time gap between them. More precisely, a temporal precedence property ϕ is written as $\phi = P_1 \prec_b P_2$, where P_1 and P_2 are (possibly guarantee) predicates and b is a positive real number. The property $\phi = P_1 \prec_b P_2$ is satisfied by a particular trajectory ξ of $\mathcal{A}$ iff

$$\forall t_2 > 0, \text{if } P_2(\xi(t_2)) \text{ then } \exists t_1, 0 < t_1 < t_2 - b, P_1(\xi(t_1)). \tag{2}$$

In other words, along ξ, predicate P_1 should be should be satisfied at least b time units *before* any instance of P_2 is satisfied. A switched system $\mathcal{A}$ satisfies ϕ, if every trajectory of $\mathcal{A}$ satisfies ϕ. With a collection of precedence properties, it is possible to state requirements about ordering of some predicates before others.

The property ϕ is said to be *robustly satisfied* by a system if $\exists \tau > 0, \delta > 0$ such that all $\tau' < \tau$ left shifts and all δ-perturbations of all trajectories ξ satisfy the property. An execution ξ is said to *robustly violate* a precedence property $P_1 \prec_b P_2$ if there is a time instant t_2 such that $P_2(\xi(t_2))$ holds, and for some $\delta > 0$, all δ-perturbations ξ' of ξ and $t_1 \in (0, t_2 - b)$, P_1 does not hold in ξ' at time t_1. A system is said to *robustly violates* $\phi = P_1 \prec_b P_2$ if some execution ξ (from an initial state) robustly violates ϕ.

3 Simulation-based Verification of Temporal Precedence

This section presents an algorithm for verifying temporal precedence properties of switched systems and establish its correctness. Similar to the simulation-based safety verification algorithm presented in an earlier work [4], this algorithm has three key features: (a) it uses validated simulations for the dynamics in $\mathcal{F}$, (b) it requires model annotations called discrepancy functions for the dynamics in in $\mathcal{F}$. Finally, (c) it requires a procedure for checking satisfiability of nonlinear guarantee predicates arising from solutions of differential equations.

For a given initial state x_0 and an ODE $\dot{x} = f(x,t)$ which admits a solution ξ, a fixed time-step numerical integrator produces a sequence of sample points e.g., $x_1, x_2, \ldots, x_l \in \mathbb{R}^n$ that approximate the trajectory ξ_{x_0} at a sequence of time points, say $\xi_{x_0}(h), \xi_{x_0}(2h), \ldots, \xi_{x_0}(l \times h)$. However, these simulations do not provide any rigorous guarantees about the errors incurred during numerical approximations. Rigorous error bounds on these simulations, which can be made arbitrarily small, are required for performing formal analysis. One such notion of a simulation for an ODE is defined as follows.

Definition 2. *Consider an ODE $\dot{x} = f(x,t)$. Given an initial state, x_0, a time bound $T > 0$, error bound $\epsilon > 0$, and a time step $\tau > 0$, an (x_0, T, ϵ, τ)-simulation trace is a finite sequence $(R_1, [t_0, t_1]), (R_2, [t_1, t_2]), \ldots, (R_l, [t_{l-1}, t_l])$ where each $R_j \subseteq \mathbb{R}^n$, and $t_j \in \mathbb{R}_{\geq 0}$ such that $\forall j, 1 \leq j \leq l$*

(1) $t_{j-1} < t_j$, $t_j - t_{j-1} \leq \tau, t_0 = 0,$ and $t_l = T$,
(2) $\forall t \in [t_{j-1}, t_j], \xi_{x_0}(t) \in R_j$, and
(3) $dia(R_j) \leq \epsilon$.

Numerical ODE solvers such as CAPD[3] and VNODE-LP [4] can be used to generate such simulations for arbitrary values of τ and ϵ using Taylor Models and interval arithmetic.

Definition 3. *A smooth function $V : \mathbb{R}^{2n} \to \mathbb{R}_{\geq 0}$ is called a* discrepancy function *for an ODE $\dot{x} = f(x,t)$, if and only if there are functions $\underline{\alpha}, \overline{\alpha} \in \mathcal{K}_\infty$ and a uniformly continuous function $\beta : \mathbb{R}^{2n} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$ with $\beta(x_1, x_2, t) \to 0$ as $|x_1 - x_2| \to 0$ such that for any pair of states $x_1, x_2 \in \mathbb{R}^n$:*

$$\underline{\alpha}(|x_1 - x_2|) \leq V(x_1, x_2) \leq \overline{\alpha}(|x_1 - x_2|) \text{ and} \tag{3}$$
$$\forall\, t > 0.\ V(\xi_{x_1}(t), \xi_{x_2}(t)) \leq \beta(x_1, x_2, t), \tag{4}$$

[3] http://capd.ii.uj.edu.pl/index.php
[4] http://www.cas.mcmaster.ca/~nedialk/vnodelp

where ξ denotes the solution of the differential equation. A tuple $(\underline{\alpha}, \overline{\alpha}, \beta)$ satisfying the above conditions is called a witness *to the discrepancy function.*

The discrepancy function provides an upper bound on the distance between two trajectories starting from different initial states x_1 and x_2. This upper bound, together with a simulation, is used to compute an overapproximation of the set of all reachable states of the system from a neighborhood of the simulation. For linear and affine dynamics such discrepancy functions can be computed by solving semidefinite programs [4]. In [4], classes of nonlinear ODEs were identified for which Lipschitz constants, contraction metrics, and incremental Lyapunov functions can be computed. These classes are all special instances of Definition 3. For the switched systems $\mathcal{A}$ with a set of differential equations $\mathcal{F} = \{f_i\}_{i \in \mathcal{I}}$, a discrepancy function for each f_i (namely, V_i and its witness $(\underline{\alpha}_i, \overline{\alpha}_i, \beta_i)$) is required. Using discrepancy function and validated simulations as building blocks, a bounded overapproximation of the reachable set for initial set Θ, set of switching signals S, and time step τ can be defined as follows.

Definition 4. *Given an initial set of states Θ, switching interval sequence S, dynamics $\mathcal{F}$, time step $\tau > 0$, and error bound $\epsilon > 0$, a $(\Theta, S, \epsilon, \tau)$-$ReachTube$ is a sequence $\psi = (O_1, [t_0, t_1]), (O_2, [t_1, t_2]), \ldots, (O_l, [t_{l-1}, t_l])$ where O_j is a set of pairs (R, h) such that $R \subseteq \mathbb{R}^n$, and $h \in \mathcal{I}$, such that, $\forall j, 1 \leq j \leq l$*

(1) $t_{j-1} < t_j, t_j - t_{j-1} \leq \tau, t_0 = 0$,
(2) $\forall x_0 \in \Theta, \forall \sigma \in sig(S), \forall t \in [t_{j-1}, t_j], \exists (R, h) \in O_j$, *such that,* $\xi_{x_0,\sigma}(t) \in R, \sigma(t) = h$,
(3) $\forall (R, h) \in O_j, dia(R) \leq \epsilon$, *and*
(4) each mode in $\mathcal{I}$ occurs at most once in O_j.

Intuitively, for every given time interval $[t_{j-1}, t_j]$, the set O_j contains an (R, h) pair such that R overapproximates the reachable set for the mode h in the given interval duration. In a previous work on verification using simulations [4], an algorithm that computes overapproximation of the reachable set via sampled executions and annotations is presented. The procedure, called ComputeReachTube, takes as input the initial set Θ, switching signals S, partitioning parameter δ, simulation error ϵ', and time step τ. It compute the sequence ψ and error ϵ such that ψ is a $(\Theta, S, \epsilon, \tau)$-$ReachTube$. The procedure is outline below.

1. Assign to Q, the set of initial states Θ.
2. For each q_i in the switching interval sequence $S = q_0, q_1, \ldots, q_k$.
3. Compute $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$, a δ-partitioning of Q, such that $Q \subseteq \cup B_\delta(x_i)$.
4. Generate a validated simulation (Definition 2) η for every state $x \in \mathcal{X}$ with error ϵ', time step τ. Then, compute the $ReachTube$ for $B_\delta(x_0)$ by bloating η as $B_\epsilon^{V_{q_i.mode}}(\eta)$, where $\epsilon = sup\{\beta_{q_i.mode}(y, x, t) | y \in B_\delta(x)\}$.
5. Compute the union of each of the $ReachTubes$ for $B_\delta(x_0)$ as the $ReachTube$ for mode $q_i.mode$.
6. Compute the initial set for the next mode by taking the projection of $ReachTube$ for $q_i.mode$ over the interval $q_i.range$ as Q. Repeat steps 3 - 6 for q_{i+1}.

The order of overapproximation of the $ReachTube$ computed using the procedure described above is the *maximum bloating* performed using the annotation $V_{q_i.mode}$ and $\beta_{q_i.mode}$ for all the modes in S. This overapproximation and the error in simulation gives the value of ϵ such that ψ is a $(\Theta, S, \epsilon, \tau)$-$ReachTube$. The nondeterminism during the switching times from one mode to another enables the reachable set to be in two different modes at a given instance of time, which is reflected in O_j. The following proposition holds [4].

Proposition 1. *Given an initial set Θ, switching signals S, partitioning parameter δ, simulation error ϵ' and time step τ, let $\langle \psi, \epsilon \rangle = \mathsf{ComputeReachTube}(\Theta, S, \delta, \epsilon', \tau)$. As $dia(\Theta) \to 0$, $width(S) \to 0$, $\delta \to 0$, $\epsilon' \to 0$, and $\tau \to 0$, then $\epsilon \to 0$.*

3.1 Temporal Precedence Verification Algorithm

CheckRefine (see Figure 1) performs the following steps iteratively: (1) Create an initial partition of the set of start states Θ. (2) Compute the $ReachTubes$ for each these partitions as given in Definition 4. (3) Check the temporal precedence property for the $ReachTube$. (4) Refine the partitioning if the above check is inconclusive, and repeat steps (2)-(4).

A key step in the procedure is to verify whether a given $ReachTube$ satisfies a temporal precedence property. The collection of intervals $mustInt$, $notInt$, and $mayInt$ for a given predicate P and $ReachTube$ ψ are used in the verification of temporal precedence properties. They are defined as follows.

Definition 5. *Given a $ReachTube$ $\psi = (O_1, [t_0, t_1]), \ldots, (O_l, [t_{l-1}, t_l])$ and a predicate P, for all $j > 0$,*

$$[t_{j-1}, t_j] \in mustInt(P, \psi) \text{ iff } \forall (R, h) \in O_j, R \subseteq P.$$
$$[t_{j-1}, t_j] \in notInt(P, \psi) \text{ iff } \forall (R, h) \in O_j, R \subseteq P^c.$$
$$[t_{j-1}, t_j] \in mayInt(P, \psi) \text{ otherwise.}$$

Definition 5 classifies an interval $[t_{j-1}, t_j]$ as an element of $mustInt(P, \psi)$ only if the overapproximation of the reachable set for that interval is contained in P. Similar is the case with $notInt(P, \psi)$. However if the overapproximation of the reachable set cannot conclude either of the cases, then the interval is classified as $mayInt(P, \psi)$. There are two possible reasons for this: first, the order of overapproximation is too coarse to prove containment in either P or P^c; second, the execution moves from the states satisfying P to states not satisfying P during that interval. Thus, better estimates of $mustInt$, $notInt$ and $mayInt$ can be obtained by improving the accuracy of $ReachTube$ ψ.

To compute $mustInt$, $mayInt$, and $notInt$ as defined in Definition 5, it is necessary to check if $R \subseteq P$ or $R \subseteq P^c$. However, for guarantee predicates with lookahead functions that use the solutions of ODEs, it is unclear how to perform these checks. Section 3.2 describes a simulation-based method to address this challenge. The algorithm in Section 3.2 will, in fact, provide weaker guarantees. Assuming P is an open set, the algorithm will answer correctly when

$R \subseteq P$ and when for some $\delta > 0$, $B_\delta(R) \subseteq P^c$; in other cases, the algorithm may not terminate. Such weaker guarantees will turn out to be sufficient for the case study considered in this paper.

Definition 6. *Given $ReachTube$ ψ and temporal precedence property $P_1 \prec_b P_2$, ψ is said to* satisfy *the property iff for any interval I', $I' \in mustInt(P_2, \psi) \cup mayInt(P_2, \psi)$, exists interval I, $I \in mustInt(P_1, \psi)$ such that $I < I' - b$. Also, ψ is said to* violate *the property if $\exists I' \in mustInt(P_2, \psi)$ such that, $\forall I \in mustInt(P_1, \psi) \cup mayInt(P_1, \psi)$, $I' - b < I$.*

From Definition 6 it is clear that if a $ReachTube$ ψ satisfies a temporal precedence property, then for all the trajectories corresponding to the $ReachTube$, the predicate P_1 is satisfied at least b time units before P_2. Also, if the $ReachTube$ violates the property, then it is clear that there exists at least one trajectory such that for an instance of time, i.e., in $I' \in mustInt(P_2, \psi)$ at all the time instances at least b units before, the predicate P_1 is not satisfied. In all other cases, the $ReachTube$ cannot infer whether the property is satisfied or violated. As this inference depends on the accuracy of $mustInt$, $notInt$ and $mayInt$. More accurate $ReachTubes$ produce better estimates of these intervals and hence help in better inference of temporal precedence property.

Given a system $\mathcal{A}$ and property $P_1 \prec_b P_2$, one can compute the $ReachTube$ for the system and apply Definition 6 to check whether the system satisfies the temporal precedence property. This is however not guaranteed to be useful as the approximation of $ReachTube$ computed might be too coarse. The algorithm CheckRefine refines, at each iteration, the inputs to compute more precise $ReachTubes$. Proposition 1 guarantees that these $ReachTubes$ can be made arbitrarily precise.

The algorithm (in Figure 1) first partitions the initial set into δ-neighborhoods (line 4) and compute $ReachTubes$ for every switching interval sequence in Ω (line 7). If all these $ReachTubes$ (that is all the executions from neighborhood) satisfy the property, then the neighborhood is removed from Q. Similarly, the algorithm CheckRefine returns that the property is violated only when $ReachTube$ violates the property. If neither can be inferred, then the parameters to function ComputeReachTube are refined in line 11 to increase their precision. Since this operation is iteratively performed to obtain arbitrarily precise $ReachTubes$, Soundness and Relative completeness follow from Definition 6 and Proposition 1.

Theorem 1 (Soundness and Relative Completeness). *Algorithm* CheckRefine *is sound, i.e., if it returns that the system satisfies the property, then the property is indeed satisfied. If it returns that the property is violated, then the property is indeed violated by the system. Further, if predicates P_1 and P_2 are open sets, and there is a procedure that correctly determines if for a set R, $R \subseteq P_i$ (for $i = 1, 2$) or if there is $\delta > 0$ such that $B_\delta(R) \subseteq P_i^c$ (for $i = 1, 2$). Then, if the system $\mathcal{A}$ satisfies $P_1 \prec_b P_2$ or robustly violates $P_1 \prec_b P_2$ then* CheckRefine *terminates with the right answer.*

```
1: Input: $\mathcal{A} = \langle \Theta, \mathcal{F}, \Sigma \rangle$, $\{V_i, (\underline{\alpha}_i, \overline{\alpha}_i, \beta_i)\}_{i \in \mathcal{I}}$, $P_1 \prec_b P_2$, $\delta_0$, $\delta'_0$, $\epsilon'_0$, $\tau_0$.
2: $Q \leftarrow \Theta$; $\Omega \leftarrow \{\Sigma\}$; $\delta \leftarrow \delta_0$; $\delta' \leftarrow \delta'_0$; $\epsilon' \leftarrow \epsilon'_0$; $\tau \leftarrow \tau_0$
3: while $Q \neq \emptyset$ do
4:   $\mathcal{X} \leftarrow \delta\text{-}partition(Q)$;
5:   for all $x_0 \in \mathcal{X}$ do
6:     for all $S \in \Omega$ do
7:       $\langle \psi, \epsilon \rangle = \mathsf{ComputeReachTube}(B_\delta(x_0), S, \delta', \epsilon', \tau)$
8:       if $\psi$ satisfies $P_1 \prec_b P_2$ then continue;
9:       else if $\psi$ falsifies $P_1 \prec_b P_2$ return "Property $P_1 \prec_b P_2$ is violated"
10:      else
11:        $\Omega \leftarrow \Omega \setminus \{S\} \cup refine(S)$; $\delta \leftarrow \delta/2$; $\delta' \leftarrow \delta'/2$, $\epsilon' \leftarrow \epsilon'/2$; $\tau \leftarrow \tau/2$;
12:        goto Line 4
13:      end if
14:    end for
15:    $Q \leftarrow Q \setminus B_\delta(x_0)$
16:  end for
17: end while
18: return "Property $P_1 \prec_b P_2$ is satisfied".
```

Fig. 1. Algorithm CheckRefine: Partitioning and refinement algorithm for verification of temporal precedence properties.

3.2 Verification of Guarantee Predicates

As discussed in the Section 2.2, guarantee predicates are of the form $P(x) = \exists t > 0, f_p(x,t) > 0$, where f_p is called a lookahead function. Section 3.1 presents an algorithm for time bounded verification of such predicates of the special form $P(x) = \exists 0 < t < T_l, w_p(\xi'_x(t)) > 0$, where w_p is a continuous function and ξ' is solution of ODE $\dot{y} = g(y,t)$. The algorihm CheckGuarantee in Figure 2 checks whether $R \subseteq P$ or an open cover of R is contained in P^c has been defined. This algorithm, similar to CheckRefine, computes successively better approximations for the $ReachTube$ and checks whether the predicate $P' \equiv w_p(x) > 0$ is satisfied by the reach tube. This is done by calculating $mustInt(P', \psi)$ and $mayInt(P', \psi)$ as defined in Definition 5. If the $mustInt$ is non-empty, then it implies that the predicate P is satisfied by the $ReachTube$ and hence $R \subseteq P$. If both the $mayInt$ and $mustInt$ are empty sets, then, clearly the predicate P is not satisfied in the bounded time T_l by any state in R, and hence an open cover of R is contained in P^c. Soundness and Relative Completeness of CheckGuarantee follow from CheckRefine (proofs in full version[5]).

4 Case Study: A Parallel Landing Protocol

The Simplified Aircraft-based Paired Approach (SAPA) is an advanced operational concept proposed by the US Federal Aviation Administration (FAA) [5].

[5] https://wiki.cites.illinois.edu/wiki/display/MitraResearch/Verification+of+a+Parallel+Landing+Protocol

```
1: Input: $R$, $\dot{y} = g(y,t)$, $S'$, $V_g(x_1,x_2)$, $(\underline{\alpha}_g, \overline{\alpha}_g, \beta_g)$ $w_p$, $\delta$, $\tau$, $T_l$
2: while $R \neq \emptyset$ do
3:     $\mathcal{X} \leftarrow \delta\text{-}partition(R)$;
4:     for all $x_0 \in \mathcal{X}$ do
5:         $\langle \psi, \epsilon \rangle = \mathsf{ComputeReachTube}(B_\delta(x_0), S', \delta, \delta, \tau)$;
6:         if $mustInt(w_p, \psi) \neq \emptyset$ then $R \leftarrow R \setminus B_\delta(x_0)$
7:         else if $mustInt(w_p, \psi) \cup mayInt(w_p, \psi) = \emptyset$ then return "UNSAT"
8:         end if
9:     end for
10:    $\delta \leftarrow \delta/2$; $\tau \leftarrow \tau/2$;
11: end while
12: return "SAT".
```

Fig. 2. Algorithm CheckGuarantee: Decides whether a lookahead predicate is satisfied in a given set R

The SAPA concept supports dependent, low-visibility parallel approach operations to runways with lateral spacing closer than 2500 ft. A Monte-Carlo study conducted by NASA has concluded that the basic SAPA concept is technically and operationally feasible [5]. SAPA relies on an alerting mechanism to avoid aircraft blunders, i.e., airspace situations where an aircraft threats to cross the path of another landing aircraft.

NASA's Adjacent Landing Alerting System (ALAS) is an alerting algorithm for the SAPA concept [10]. ALAS is a pair-wise algorithm, where the two aircraft are referred to as *ownship* and *intruder*. When the ALAS algorithm is deployed in an aircraft following the SAPA procedure, the aircraft considers itself to be the ownship, while any other aircraft is considered to be an intruder. The alerting logic of the ALAS algorithm consists of several checks including conformance of the onwship to its nominal landing trajectory, aircraft separation at current time, and projected aircraft separation for different trajectories.

A formal static analysis of the ALAS algorithm is challenging due to the complexity of the SAPA protocol and the large set of configurable parameters of the ALAS algorithm that enable different alerting thresholds, aircraft performances, and runway geometries. This paper considers the component of the ALAS alerting logic that checks violations of predefined separation minima for linear and curved projected trajectories of the current aircraft states. This component is one of the most challenging to analyze since it involves nonlinear dynamics. Safety considerations regarding communication errors, pilot and communication delays, surveillance uncertainty, and feasibility of resolution maneuvers are not modeled in this paper.

For the analysis of the landing protocol, this paper considers a blundering scenario where the intruder aircraft turns towards the ownship during the landing approach. The dynamics of the aircraft are modeled as a switched system with continuous variables sx_i, sy_i, vx_i, vy_i and sx_o, sy_o, vx_o, and vy_o representing the position and velocity of intruder and ownship respectively. The switching system has two modes: $approach$ and $turn$. The mode $approach$ represents the phase when both aircraft are heading towards the runway with constant speed. The mode $turn$ represents the blundering trajectory of intruder. In this mode, the intruder banks at an angle ϕ_i to turn away from the runway towards the ownship. The switching signal determines the time of transition from $approach$ to $turn$. In this mode, the differential equation of the ownship remains the same as that of $approach$, but the intruder's turning motion with banking angle ϕ_i is

$$\dot{\begin{bmatrix} sx_i \\ sy_i \\ vx_i \\ vy_i \end{bmatrix}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \omega_i \\ 0 & 0 & -\omega_i & 0 \end{bmatrix} \begin{bmatrix} sx_i \\ sy_i \\ vx_i \\ vy_i \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega_i - c_y \\ \omega_i + c_x \end{bmatrix}, \tag{5}$$

where c_x and c_y are constant functions of the initial states of the ownship and intruder, and ω_i is the angular speed of intruder. Given the bank angle ϕ_i, the angular speed is given by $w_i = \frac{G|\tan(\phi_i)|}{\sqrt{vx_i^2 + vy_i^2}}$, where G is the gravitational constant. The upper bound on the bank angle ϕ_i is denoted as ϕ_{max}.

The system starts in the $approach$ mode with the initial position of the intruder at $sx_i = sy_i = 0$ and the ownship at $sx_o = xsep$ and $sy_o = ysep$, where $xsep$ denotes the lateral separation between the runways and $ysep$ denotes the initial longitudinal separation between the aircraft. The initial velocities of both aircraft along the x-axis are 0 and the initial velocities along the y-axis are parameters. The time of switching from $approach$ mode to $turn$ mode is nondeterministically chosen from the interval $T_{switch} = [2.3, 2.8]$. These parameters and the initial values of the variables are constrained by the SAPA procedure [5].

4.1 Alerting Logic and Verification of Temporal Precedence Property

The alerting logic of ALAS considered in this paper issues an alert when the aircraft are predicted to violate some distance thresholds called $Front$ and $Back$ [10]. To predict this violation, the aircraft projects the current state of the system with three different dynamics: first, the intruder does not turn, i.e., banking angle $0°$, second, the intruder turns with the specified bank angle ϕ_i and third, the intruder turns with the maximum bank angle ϕ_{max}. If any of these projections violates the distance thresholds, then an alert is issued. The alert predicates for the each one of these projections are represented by $Alert_0$, $Alert_{\phi_i}$ and $Alert_{\phi_{max}}$, respectively. Thus, the alerting logic considered in this paper is defined as $Alert \equiv Alert_0 \vee Alert_{\phi_i} \vee Alert_{\phi_{max}}$.

The alert predicates $Alert_0$, $Alert_{\phi_i}$ and $Alert_{\phi_{max}}$ are guarantee predicates. The lookahead function for $Alert_\pi$ is defined as follows: from a given state x, it computes the projected trajectory of the aircraft when intruder turns at bank angle π. If these trajectories intersect, then it computes the times of intersection.

That is, it computes t_i, t_o such that $sx'_i(t_i) = sx'_o(t_o)$ and $sy'_i(t_i) = sy'_o(t_o)$, where $sx'_i, sy'_i, sx'_o, sy'_o$ represent the positions of the intruder and ownship aircraft in the projected trajectory. If such t_i and t_o exist, the $Alert_\pi$ is defined as:

$$\begin{aligned} Alert_\pi(x) \equiv \text{ iff } t_i > t_0 \text{ ? } & (\Delta t^2 \times (vx_o^2 + vy_o^2) < Back^2) \\ : \ & (\Delta t^2 \times (vx_o^2 + vy_o^2) < Front^2), \end{aligned}$$

where $\Delta t = t_i - t_o$. If such t_i and t_o do not exist, then $Alert_\pi(x) = \bot$. The expression $a \mathbin{?} b \ : c$ is a short hand for **if**(a) **then** b **else** c.

As the guarantee predicates cannot be handled by SMT solvers, Section 3.2 proposes a simulation based algorithm for handling them. In this case study, the proposed technique is used to resolve the nonlinearities of t_o and t_i in the $Alert_\pi$ predicate. An overapproximation $Alert'_\pi$ of $Alert_\pi$ is computed as: $Alert'_\pi(x) = \top$ if and only if

$$\begin{aligned} T_i > T_0 \text{ ? } & (\Delta T^2 \times (vx_o^2 + vy_o^2) < Back^2) \\ : \ & (\Delta T^2 \times (vx_o^2 + vy_o^2) < Front^2) \end{aligned}$$

where $\Delta T = T_i - T_o$. The numerical values of T_i and T_o computed simplify the $Alert'_\pi$ predicate and can be handled by SMT solvers.

A state of the system where the intruder aircraft is inside a safety area surrounding the ownship is said to be *unsafe*. This paper considers a safety area of rectangular shape that is $SafeHoriz$ wide, starts a distance $SafeBack$ behinds the ownship and finishes a distance $SafeFront$ in front of the ownship. The values $SafeHoriz$, $SafeBack$ and $SafeFront$ are given constants. Formally, the predicate $Unsafe$ is defined as $Unsafe(x) \equiv (sy_i > sy_o ? sy_i - sy_o < SafeFront : sy_o - sy_i < SafeBack)$ and $|sx_i - sx_o| < SafeHoriz$.

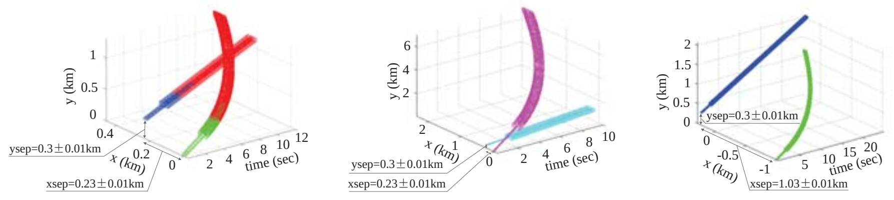
The correctness property considered in this paper is that an alert is raised at least b seconds before the intruder violates the safety buffer where b is in the range $[4, 15]$. This can written as a temporal precedence property $Alert \prec_b Unsafe$.

4.2 Verification Scenarios and C2E2 Performance

The verification algorithms of Section 3 are implemented in the tool Compute Execute Check Engine (C2E2). C2E2 accepts Stateflow (SF) charts as inputs, translates them to C++ using CAPD for generating rigorous simulations. For checking SAT queries, it uses Z3 [1] and GLPK[6]. The discrepancy functions for the aircraft dynamics were obtained by computing incremental Lyapunov-like function using MATLAB [4]. The following experiments were performed on Intel Quad Core machine 2.33 GHz with 4GM memory.

The temporal precedence property $Alert \prec_b Unsafe$ is checked for several configurations of the system, i.e., values of parameters and initial values of state variables. For these experiments, the time bound for verification is set to 15 seconds and the time bound for projection is set to 25 seconds.

[6] http://www.gnu.org/software/glpk



(a) Scenario 1. (b) Scenario 2. (c) Scenario 3.

Fig. 3. Figure depicting the set of reachable states of the system. Color coding is used to depict whether the alert is issued by the alerting algorithm

Scenario 1. The system configuration is specified by the following parameters and variables: $xsep \in [0.22, 0.24]$ km, $ysep \in [0.2, 04]$ km, $\phi_i = 30°$, $\phi_{max} = 45°$, $vy_o = 0.07$ km/s and $vy_i = 0.08$ km/s. With this configuration, C2E2 proves that the system satisfies the temporal precedence property $Alert \prec_4 Unsafe$ and an alert is generated 4.38 seconds before the safety is violated. The set of reachable states of the ownship and the intruder when the safety property is violated is shown in *red* and the safe states reached are shown in blue and green respectively in Figure 3(a).

Scenario 2. Increasing the intruder velocity to $vy_i = 0.11$ km/s, and bank angle $\phi_i = 45°$ from the configuration of Scenario 1 results in Scenario 2. In this case, the safe separation between the intruder and the ownship is always maintained as the intruder completes the turn behind the ownship. Also, the alarm is not raised and hence the property $Alert \prec_b Unsafe$ is satisfied.

Scenario 3. Changing the configuration by $vy_i = 0.11$ km/s, $xsep \in [1.02, 1.04]$ km, and $\phi_i = 45°$ from Scenario 1 results in Scenario 3. C2E2 proves that the simplified alerting logic considered in this paper issues a false-alert, i.e., an alert is issued even when the safety of the system is maintained. Though the property $Alert \prec_b Unsafe$ is not violated, avoiding such circumstances improves the efficiency of the protocol and C2E2 can help identify such configurations.

Scen.	$A \prec_4 U$	time (m:s)	Refs.	$A \prec_t U$
6	False	3:27	5	2.16
7	True	1:13	0	–
8	True	2:21	0	–
6.1	False	7:18	8	1.54
7.1	True	2:34	0	–
8.1	True	4:55	0	–
9	False	2:18	2	1.8
10	False	3:04	3	2.4
9.1	False	4:30	2	1.8
10.1	False	6:11	3	2.4

Table 1. Running times. Columns 2-5: Verification Result, Running time, # of refinements, value of b for which $A \prec_b U$ is satisfied.

Scenario 4. Placing the intruder in front of ownship, i.e., $ysep = -0.3$ km and $vy_i = 0.115$ km/s from configuration in Scenario 1 results in Scenario 4. C2E2 proves that the simplified alerting logic considered in this paper misses an alert, i.e., does not issue an alert before the safety separation is violated. Such scenarios should always be avoided as they might lead to catastrophic situations. This demonstrates that C2E2 can aid in identifying scenarios which should be avoided and help design the safe operational conditions for the protocol.

Scenario 5. Reducing the $xsep \in [0.15, 0.17]$ km and $ysep \in [0.19, 0.21]$ km from configuration in Scenario 1 gives Scenario 5. For this scenario, C2E2 did not terminate in 30 mins. Since the verification algorithm presented in Section 3 is sound and relatively complete only if the system robustly satisfies the property, it is conjectured that Scenario 5 does not satisfy the property robustly. Upon closer inspection, it is observed that the partitioning parameter $\delta = 0.0005$ and time step $\tau = 0.001$ (typical values at termination are $\delta = 0.005$ and $\tau = 0.01$), which support the conjecture.

The running time of verification procedure and their outcomes for several other scenarios are presented in Table 1. Scenarios 6-8 introduce uncertainty in the initial velocities of the aircraft with all other parameters remaining the same as in Scenario 1. The velocity of the aircraft are changed to be $vy_o \in [0.07, 0.075]$ in scenario 5, $vy_i \in [0.107, 0.117]$ in scenario 6, and $vx_i \in [0.0, 0.005]$ in scenario 7 respectively. Scenarios $x.1$ is similar to Scenario x (for x being 6,7,8), however with twice the uncertainty in the velocity. Scenario 9 is obtained by changing the runway separation to be $xsep = 0.5 \pm 0.01$. Scenario 10 is obtained by reducing the $xsep = 0.2 \pm 0.01$. Scenario $x.1$ is similar to Scenario x (for x being 9,10) however with twice the time horizon for verification and projection. These results suggest that the verification time depends on time horizon approximately linearly.

5 Related Work and Conclusion

There are several MATLAB based tools for analyzing properties of switched systems using simulations. Breach [3] uses sensitivity analysis for analyzing STL properties of systems using simulations. This analysis is sound and relatively complete for linear systems, but does not provide formal guarantees for nonlinear systems. S-Taliro [9] is a falsification engine that search for counterexamples using Monte-Carlo techniques and hence provides only probabilistic guarantees. STRONG [2] uses robustness analysis for coverage of all executions from a given initial set by constructing bisimulation functions. Currently this tool computes bisimulation functions for only linear or affine hybrid systems and does not handle nonlinear systems.

This paper presents a dynamic analysis technique that verifies *temporal precedence properties* and an approach to verify guarantee predicates that use solutions of ODEs as lookahead functions. These techniques are proved to be sound and relative complete. The verification approach is applied to a landing protocol that involves nonlinear dynamics. The case study demonstrated that the proposed technique can not only verify safety properties of the alerting logic, but also could identify conditions for false and missed alert which are crucial in designing the operational concept.

References

1. Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340, Budapest, Hungary, 2008. Springer.
2. Yi Deng, Akshay Rajhans, and A. Agung Julius. STRONG: A trajectory-based verification toolbox for hybrid systems. In *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST 2013)*, volume 8054 of *Lecture Notes in Computer Science*, pages 165–168, Buenos Aires, Argentina, 2013. Springer.
3. Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Proceedings of the 22nd International Conference on Computer Aided Verification (CAV 2010)*, volume 6174 of *Lecture Notes in Computer Science*, pages 167–170, Edinburgh, UK, 2010. Springer.
4. Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Verification of annotated models from executions. In *Proceedings of the 13th International Conference on Embedded Software (EMSOFT 2013)*, Montreal, Canada, 2013.
5. Sally C. Johnson, Gary W. Lohr, Burnell T. McKissick, Nelson M. Guerreiro, and Paul Volk. Simplified aircraft-based paired approach: Concept definition and initial analysis. Technical Report NASA/TP-2013-217994, NASA, Langley Research Center, 2013.
6. Daniel Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhauser, Boston, June 2003.
7. W. Lohmiller and J. J. E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 32(6):683–696, 1998.
8. Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC '87)*, pages 205–205, Vancouver, British Columbia, Canada, 1987. ACM.
9. Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivancić, Aarti Gupta, and George J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC 2010)*, pages 211–220, Stockholm, Sweden, 2010. ACM.
10. Raleigh B. Perry, Michael M. Madden, Wilfredo Torres-Pomales, and Ricky W. Butler. The simplified aircraft-based paired approach with the ALAS alerting algorithm. Technical Report NASA/TM-2013-217804, NASA, Langley Research Center, 2013.
11. G.R. Wood and B.P. Zhang. Estimation of the Lipschitz constant of a function. *Journal of Global Optimization*, 8:91–103, 1996.